Parametric Feature Detection

CUCS-028-95

Shree K. Nayar, Simon Baker, and Hiroshi Murase

Department of Computer Science
Columbia University
New York, NY 10027

Abstract

A large number of visual features are parametric in nature, including, edges, lines, corners, and junctions. We present a general framework for the design and implementation of detectors for parametrized features. For robustness, we argue in favor of elaborate modeling of features as they appear in the physical world. In addition, optical and sensing artifacts are incorporated to achieve realistic feature models in image domain. Each feature is represented as a densely sampled parameterized manifold in a low-dimensional subspace. During detection, the brightness distribution around each image pixel is projected to the subspace. If the projection lies close to the feature manifold, the exact location of the closest manifold point reveals the parameters of the feature. The concepts of parameter reduction by normalization, dimension reduction, pattern rejection, and efficient search are employed to achieve high efficiency.

Detectors have been implemented for five specific features, namely, step edge (5 parameters), roof edge (5 parameters), line (6 parameters), corner (5 parameters), and circular disc (6 parameters). All five of these detectors were generated using the same technique by simply inputing different feature models. Detailed experiments are reported on the robustness of detection and the accuracy of parameter estimation. In the case of the step edge, our results are compared with those obtained using popular detectors. We conclude with a brief discussion on the use of relaxation to refine outputs from multiple feature detectors, and sketch a hardware architecture for a general feature detection machine.

Category: Low-Level Processing, Pattern Analysis

Index Terms: Parametrized features, feature modeling, optical effects, sensor effects, feature manifolds, normalizations, dimension reduction, efficient search, feature detection, parameter estimation, step edges, roof edges, corners, lines, discs, software modules, relaxation, modular hardware architecture.

1 Introduction

Most applications in image processing and computational vision rely on robust detection of image features and accurate estimation of their parameters. An example of a parametrized feature is the step edge. It is by far the most intensely studied, both due to its abundance in natural scenes, and due to its simplicity. The step edge, however, is by no means the only feature of interest in image understanding. It is closely followed in significance by other ubiquitous ones such as *lines, corners, junctions,* and *roof edges*[1]. This list is far from comprehensive, even if we restrict ourselves to the class of well-defined and commonplace features. Moreover, in any fixed application, the term feature may take on a meaning that is specific to that application. For instance, in the inspection or recognition of a manufactured part, a subpart such as bolt may be the feature of interest. Such a feature may also be parametric in nature. The rotation of the head of the bolt would result in a continuum of image appearances. In short, features may be too numerous to justify the process of deriving a new detector for each one. Is it possible to develop a single detection mechanism that is applicable to *any* parametrized feature?

This is exactly the objective of our work. We seek a general and easy to implement methodology for detecting parameterized features within an image. In addition to detection, we wish to obtain precise estimates of the feature parameters. In most real applications, detection alone does not suffice. Feature parameters, if recovered with precision, can be of vital importance to higher levels of visual processing. A simple example is that of the generalized Hough transform, where accurate knowledge of edge direction reduces the dimension of the Hough space by one. Likewise, the performance of any boundary growing algorithm is dramatically enhanced when the directions of image edgels are used to guide the growth of the boundary. In our framework, detection and parameter estimation go hand in hand. The end result is a rich yet concise description of an image.

If one is in pursuit of high performance in both detection and parameter estimation, it is essential to accurately model the features as they appear in the physical world. We do this by defining highly descriptive models of features that include as many parameters as necessary to accurately represent the feature. We do not make simplifications for analytic or efficiency reasons. For instance, our model of the step edge has 5 parameters, namely, the lower brightness level, the brightness difference across the step, the angle (orientation) of the edge, its intrapixel location, and a blurring (scaling) parameter. Our experimental results indicate that the intrapixel localization corresponds to approximately 13^o of edge orientation in terms of representational accuracy. The blurring parameter turns out to be

[1] A concise description of the different types of features and a review of existing step edge detectors can be found in [Nalwa 93]. A survey of early edge detectors can be found in [Davis 75]. Given the extent to which feature detection has been explored, a survey of all the work in this area is well beyond the scope of this paper. In our discussion, we only use examples of previous detectors without attempting to mention all of them. Further, we will be primarily interested in examples that use parametric feature models rather than those based on the intensity gradient or other differential invariants.

almost equally important. Hence, we argue that reliable step edge detection requires the use of a model that includes both intrapixel localization and blur parameters. Conversely, edge detectors which are not based on such descriptive models are fundamentally limited. We arrive at similar conclusions for the other features we experimented with. We used a 5 parameter model for roof edges, a 6 parameter model for lines, a 5 parameter model for corners, and a 6 parameter model for circular discs.

Hitherto, our notion of a feature model has been a continuous one. Previous work on feature detection has almost always designed continuous detectors using continuous feature models and then sampled the detectors before applying them to discrete images. However, in order to obtain a reliable detector, careful consideration must be given to the conversion of the radiance function of a feature in the scene to its discrete image produced by a sensor. For instance, the aspect ratio of an image sensor clearly impacts the appearance of a feature. Perhaps less obvious are the effects of the shape and size of the photosensitive elements on a CCD image sensor. Our results show that, these effects translate to a representational error equivalent to approximately 0.5^o of orientation, say for the step edge. Though such an error may seem small, it can degrade performance in many applications. In addition, we also model the blurring caused by the optical transfer function of the imaging optics. Modeling these sensor and optical characteristics leads to very realistic descriptions of measured features. Note that, altering the sensor and optical models to match the specifications of the particular imaging system is simple. It requires the change of just one system module.

The detailed models of features and of the imaging system allow us to accurately predict the pixel intensity values in a window about the feature. All we require are the feature's parameters. If we treat the pixel values as real numbers, we can think of each parameterized feature as corresponding to a parameterized manifold in $\Re^N$, where N is the number of pixels in the window surrounding the feature. As the parameters of the feature vary, the point in $\Re^N$ corresponding to the feature, traces out a k-dimensional manifold, where k is the number of feature parameters. In this setting, feature detection means finding the closest point on the manifold to the point in $\Re^N$ corresponding to pixel values in a novel image window. If the closest point is near enough, we register the presence of the feature. The exact location of the closest manifold point reveals the parameters of the feature. If, on the other hand, the nearest manifold point is too far away, we declare the absence of the feature. This statement of the feature detection problem is certainly not new. It was first introduced for step edges and lines by Hueckel [Hueckel 71], and subsequently used by Hummel [Hummel 79] for step edges. More elaborate parametric models for step edges were used by Haralick [Haralick 84] and Nalwa and Binford [Nalwa and Binford 86], amongst others (see [Nalwa 93]).

Hueckel and Hummel both argued that, for efficiency, a closed form solution to the feature fitting problem must be found. To make their derivations possible, they used simple continuous models for step edges and lines. Our view of feature detection is radically different. We believe that the features we wish to detect are inherently complex visual entities. Hence, we willingly forego all hope of finding closed-form solutions for the best-fit parameters. Instead, we discretize the search problem by densely sampling the feature manifold. The closest point on the manifold is then approximated by finding the nearest neighbor amongst the sample points. Typically, this sampling may result in the order of 10^5 points, which lie in a space of dimension, $N = 25$–100. Further, note that the search for the closest manifold point must be repeated for each window (centered around each pixel) in the image.

At first glance, this should seem inefficient to the point of impracticality. However, we will show that our approach is in fact very practical. To obtain the required efficiency we used a number of different techniques. First, we introduce a set of simple normalization procedures that reduce the dimensionality of the manifold to 3 or 4 (for the 5 features we experimented with) without loss of information or reduction of the signal-to-noise ratio (S.N.R.). Next, we apply the Karhunen-Loéve (K-L) expansion [Oja 83], as a dimension reduction technique. This enables us to achieve high detection rates by projecting the feature manifolds into spaces of dimension, $d \ll N$. In practice, d is generally in the range 4–10. Such a compressed representation was proposed for 3-D object recognition and pose estimation in [Murase and Nayar 95].

During detection itself, we use a heuristic search algorithm that exploits the local smoothness of the manifolds, to quickly find the closest sample point. Further, it turns out that we do not need to perform the search for every pixel in the image. Amongst other rejection techniques, we use a recently developed rejection algorithm [Baker and Nayar 95] to quickly eliminate a vast majority of pixels from further consideration, without even projecting into the K-L subspace. Such a rejection scheme is feasible and effective since most pixels in an image do not represent features of interest. With the above ideas in place, our feature detectors, even in their present unoptimized implementation, take only a few seconds on a standard single-processor workstation when applied to a 512×480 image. Given the enormous strides being made in memory and multi-processor technology, it is only a matter of time before real-time performance is achieved.

This approach to feature detection and parameter estimation differs from previous ones in two significant ways:

- Our system offers a level of generality that is uncommon in the realm of feature detection. As far as we can ascertain, there is no single technique capable of detecting even the five features (step edges, lines, corners, circular discs, and roof edges) we implemented. More importantly, the addition of a new feature to our system is simply a matter of writing a single "C" function that defines the feature model. Alternatively, features could be defined by experimentally obtained data sets without any difficulty.

- Most previous approaches have used simple feature models with detection as the main goal and not parameter estimation. Such models do not entirely capture the image properties of scene features. The descriptive nature of our features models and the incorporation of sensor and optical effects give our features an unusual level of realism. This has enabled robust detection and parameter estimation for each of the complex features we implemented.

The paper is organized as follows. In the following section, we introduce the notion of a generic scene feature and discuss the modeling of imaging and sensing effects. We show how this leads to the parametric feature manifold representation. Then, the application of normalization and dimension reduction is described. In section 3, we model our five example features, namely, step edges, lines, corners, roof edges, and circular discs. In each case, the feature model, the result of dimension reduction, and the manifold representation of the feature are presented. In section 4, the implementation of detection is discussed, via manifold sampling, efficient search, and the use of rejection techniques. In section 5, our experimental results and comparisons with the Canny [Canny 86] and Nalwa-Binford [Nalwa and Binford 86] detectors are included. In section 6, we briefly describe a modular software package we are developing that will enable easy use of the proposed method. The idea of using relaxation for extracting high-level image descriptions from multiple detected features and their parameters is outlined. Finally, we briefly mention how our general feature detector lends itself to a simple but efficient hardware implementation.

2 Parametric Feature Representation

We begin by presenting the theoretical basis of our approach to feature detection. First, the notion of an arbitrary parameterized scene feature is introduced. Then, we describe the artifacts introduced by the imaging system as it maps a scene feature to its discrete image. The family of images obtained by varying feature parameters is represented as a parametric manifold in a finite dimensional Hilbert Space. Simple but important normalizations are introduced that reduce the dimensionality of the parametric manifold. Finally, a dimension reduction technique is invoked to obtain parametric feature manifolds in low-dimensional subspaces.

2.1 Parametric Scene Features

By a scene feature we mean a geometric and/or a photometric phenomenon in the physical world that produces spatial radiance variations which, if detectable, can aid in visual perception. Let us assume that the imaging system is perfect, in which case, image brightness is proportional to scene radiance. The image feature is then the continuous radiance function of the scene feature. It can be written as $F^c(x, y; \mathbf{q})$ where $(x, y) \in S$ are image points within a finite feature window, S, and $\mathbf{q}$ are the parameters of the feature. For instance, in the case of a step edge, $\mathbf{q}$ would include edge orientation and the brightness values on the two sides of the edge. In the case of a corner, it would include the orientation of the corner, the angle made by the corner, and the brightness values inside and outside the corner. Note that, to fully specify a feature, we need to provide the feature function, $F^c(x, y; \mathbf{q})$, the feature window, S, and the ranges of the parameters, $\mathbf{q}$.

2.2 Modeling Image Formation and Sensing

It is interesting to note that previous work on feature detection has overlooked the artifacts induced by the imaging system. Two reasons could serve to justify this. First, some of these artifacts are nonlinear in nature and would only make the problem of detector derivation, as approached before, more cumbersome. Second, the effects introduced by the imaging system are typically less pronounced than those that result from the feature parameters themselves. For reasons that will become clear shortly, we are able to incorporate both linear and nonlinear effects in our feature model. Hence, we have chosen to make our feature models as precise as possible by incorporating the effects of image formation.

The first effect is the blurring of the continuous feature image. If the scene feature lies outside the focused plane of the imaging system, its image is defocused. Further, the finite size of the lens aperture causes the optical transfer function of the imaging system to be bandlimited in its spatial resolution. In addition, the feature itself, even before imaging, may appear somewhat blurred. For instance, a real scene edge would not be a perfect step but rather rounded. The effect of this in image space clearly depends on the magnification of the imaging system. The defocus factor can be approximated as a pillbox function [Born and Wolf 65], the optical transfer function by the square of the first-order Bessel function of the first kind [Born and Wolf 65], and the blurring due to imperfections in the feature by a Gaussian function [Koenderink 84]. We lump all three effects in a single blurring factor that is assumed to be a 2-D Gaussian function:

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp(-\frac{1}{2} \cdot \frac{x^2 + y^2}{\sigma^2}) \tag{1}$$

The continuous image on the sensor plane is converted (typically by a CCD detector) to a discrete image through two processes. First, the light flux falling within each pixel is summed, or averaged. If the pixels are rectangular in structure [Barbe 80] [Norton 82], the averaging function is simply the rectangular function [Bracewell 78]:

$$a(x, y) = \frac{1}{w_x\, w_y} {}^2\Pi(\frac{1}{w_x} x, \frac{1}{w_y} y) \tag{2}$$

where, w_x and w_y are the x and y dimensions of the pixel, respectively.

The sampling function that converts the continuous image to a discrete one is often a rectangular grid:

$$s(x, y) = {}^2\mathrm{III}(\tfrac{1}{p_x} x, \tfrac{1}{p_y} y) \tag{3}$$

where, p_x and p_y are spacings between discrete samples in the two spatial dimensions. It is important to note that the blurring, averaging, and sampling functions can vary between sensors. Above, we have assumed the pixels and the sampling to be rectangular. In practice, these functions must be selected based on the specifications of the image sensor used. The final discrete image of a feature may now be written as:

$$F(x, y; \mathbf{q}) = \{\, F^c(x, y; \mathbf{q}) \, * \, g(x, y) \, * \, a(x, y) \,\} \, . \, s(x, y) \tag{4}$$

where, $*$ is the 2-D convolution operator. Since the above image is simply a weighted sum of Kronecker delta functions [Bracewell 78], it can also be written as $F(m, n; \mathbf{q})$, where $(m, n) \in S$ are the (integer valued) pixels coordinates of the discrete sample locations within the feature window.

2.3 Parametric Feature Manifolds

If the total number of pixels in the window is N, each feature image, $F(m, n; \mathbf{q})$, may be regarded as a point in an N-dimensional Hilbert space. Suppose the feature has k parameters ($\dim(\mathbf{q})=k$). Then, as the parameters vary over their ranges, the point traces out a k-parameter manifold in the N-dimensional Hilbert space. It is possible therefore to represent each feature of interest as a multivariate manifold in a high-dimensional space. Feature detection then entails finding the closest point of the feature manifold for each novel candidate window in the image. Clearly, this is impractical given the high dimensionality of the Hilbert space (N) and the manifold itself (k). In the following subsections, we present two techniques that dramatical reduce dimensionality, making the feature manifold a viable representation for detection and parameter estimation.

2.4 Parameter Reduction by Normalization

For each feature instance, we compute its mean $\mu(\mathbf{q}) = \frac{1}{N} \sum_{(n,m) \in S} F(m, n; \mathbf{q})$, and its magnitude $\nu(\mathbf{q}) = \|\, F(m, n; \mathbf{q}) - \mu(\mathbf{q}) \,\|$. The following brightness normalization is then applied:

$$F'(m, n; \mathbf{q}) = \frac{1}{\nu(\mathbf{q})} \, (\, F(m, n; \mathbf{q}) \; - \; \mu(\mathbf{q}) \,) \tag{5}$$

This simple normalization proves to be valuable. In all the features we have implemented, it has reduced the dimensionality of the manifold by two. This happens because, $F'(m, n; \mathbf{q})$ turns out to be (approximately) independent of two of the parameters in $\mathbf{q}$. For instance, in the case of the step edge, the normalized feature is invariant to the brightness values on either side of the step. All step edges, irrespective of their brightness values, are reduced to step edges with the same brightness on either side. It is important to note that this normalization does not alter the signal-to-noise ratio of the edge prior to normalization. The normalization is applied not only while constructing the feature manifold but also during feature detection. Once a feature has been detected, its mean μ and magnitude ν can be used to recover the two brightness parameters eliminated during normalization.

2.5 Dimension Reduction

In most parameterized features, the image $F' = F'(m, n; \mathbf{q})$ varies gradually with the parameters $\mathbf{q}$. As a result, consecutive feature instances tend to be highly correlated. Further, inherent symmetries in the structures of certain features add to the similarity between instances. It is therefore possible to represent the feature manifold in a low-dimensional subspace without significant loss of information[2]. If correlation between feature instances is the preferred measure of similarity (or dissimilarity), the Karhunen-Loéve (K-L) expansion [Oja 83] [Fukunaga 90], yields the optimal subspace.

The covariance matrix $\mathbf{R} = E[(F' - E[F'])(F' - E[F'])^T]$ represents the correlation between corresponding pixels in the different feature instances. The normalized feature instances F' are N-dimensional vectors, and so $\mathbf{R}$ is a symmetric $N \times N$ matrix. The reduced space is computed by solving the eigenstructure decomposition problem:

$$\mathbf{R}\,\mathbf{e} = \lambda\,\mathbf{e} \tag{6}$$

The result is the set of eigenvalues $\{\, \lambda_j \mid j = 1, 2, ..., N \,\}$ where $\lambda_1 \geq \lambda_2 \geq \ ..... \ \geq \lambda_N \geq 0$, and a corresponding set of orthonormal eigenvectors $\{\, \mathbf{e}_j \mid j = 1, 2, ..., N \,\}$. Due to the inherent structure of most parametrized features, the first few eigenvalues tend to be significantly larger than the remaining ones. This allows us to represent features in a low-dimensional subspace spanned by the few most prominent eigenvectors. Suppose we use the first d eigenvectors. Then, a measure of the information discarded is the K-L *residue* defined by:

$$R(d) = \sum_{j=d+1}^{N} \lambda_j \tag{7}$$

As an example, a step edge with a 49-D Hilbert space (pixels) can be represented in a 3-D subspace with K-L residue 10%. Moving to an 8-D subspace reduces the residue to 2%.

The parametric feature manifold is constructed by simply projecting all instances of the feature to the subspace. This only requires dot products of each feature instance with the prominent eigenvectors that serve as a basis for the subspace. Since such a

[2]This idea was first explored by Hummel [Hummel 79] for the case of an ideal step edge. Hummel analytically derived the optimal basis for representing a step edge. Later, a similar derivation was put forth by Lenz [Lenz 87] for the case of an ideal line. Such closed-form derivations require detailed manipulations and the use of simplistic feature models to facilitate analysis. Furthermore, even with the use of simple models, closed-form solutions may not exist for many features of interest. Our approach is to use elaborate feature models to generate feature instances for different parameter values, and then use numerical methods to compute the optimal basis. This results in higher precision and greater generality.

parameterized feature manifold is easy to compute for any feature, we have at our disposal a generic tool for designing feature detectors. Further, the dramatic dimension reduction, and the parameter reduction by brightness normalization in section 2.4, together allow us to compactly represent features and detect them efficiently.

3 Example Features

We now illustrate manifold representations of 5 parametrized features. For each feature, we provide a formal definition of the feature and its parameters, discuss the effects of brightness normalization, and present results on dimension reduction.

The features we have chosen to discuss here are merely examples that happen to be of import in machine vision. It is worth adding that the techniques we have developed are by no means restricted to features in brightness images. The same techniques may be applied to features found in data produced by other sensors, such as, infrared, X-ray, ultrasound, and range sensors.

3.1 Step Edge

Our first example feature is the familiar *step edge.* Parametric models for edges date back to the work of Hueckel [Hueckel 71]. Since then, the edge has been studied in more detail than any other visual feature (see [Davis 75][Nalwa 93]). Figures 1(a) and 1(b) show isometric and plan views of the step edge model we use. This model is a generalization of those used in [Hueckel 71], [Hummel 79], and [Lenz 87]. It is closest to the one used by Nalwa and Binford [Nalwa and Binford 86], but differs in its treatment of smoothing effects.

The basis for the 2-D step edge model is the 1-D unit step function:

$$u(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \tag{8}$$

A step with lower intensity level, A, and upper intensity level, $A + B$, can be written as $A + B \cdot u(t)$. To extend to 2-D, we assume that the step edge is of constant cross section (step size along its length), it is oriented at an angle θ, and lies at a distance, ρ, from the origin. Then, the orthogonal distance of an arbitrary point (x, y) from the step (see Figure 1(b)) is given by:

$$z = y \cdot \cos\theta - x \cdot \sin\theta - \rho \tag{9}$$

Therefore, an ideal step edge of arbitrary orientation and displacement from the origin is given by the 2-D function $A + B \cdot u(z)$. For reasons given in section 2.2, we need to incorporate the Gaussian blurring function, the pixel averaging function, and the sampling function. The final step edge model is:

$$F_{SE}(x,y;A,B,\theta,\rho,\sigma) = \{\,(A + B\cdot u(z)) * g(x,y;\sigma) * \; a(x,y)\,\}\,.\,s(x,y) \qquad (10)$$

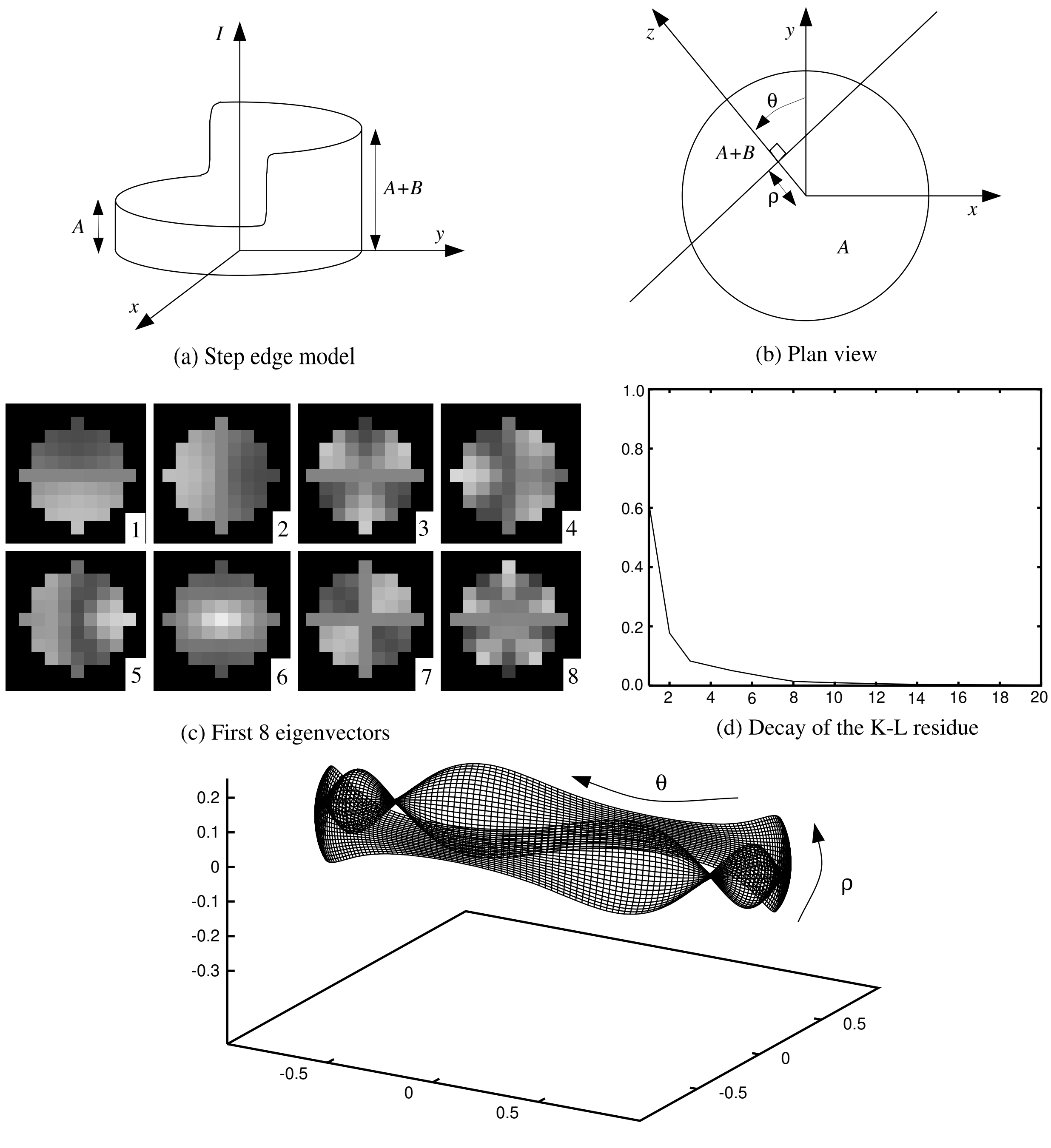
where, z is given by equation (9). Note that our edge model has 5 parameters, namely, orientation θ, localization ρ, blurring or scaling σ, and the brightness values A and B.

To complete our definition of the step edge, we need to specify ranges that the parameters may take. Here, distances are in units of the distance between two neighboring pixels, and angles in degrees. The orientation parameter, θ, is drawn from $[0^o, 360^o]$. We restrict the localization parameter, ρ, to lie in $[-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, since any edge must pass closer than a distance $\frac{\sqrt{2}}{2}$ from the center of at least one pixel in the image. The blurring parameter, σ, is drawn from $[0.3, 1.5]$. As described in [Nalwa and Binford 86], substantially larger values of σ could be used, but really represent an edge at a much higher magnification. Such cases would require the use of large image window for detection. The intensity parameters A and B are free to take any value. This is because of the normalizations described in section 2.4. The structure of a normalized step edge, given by the parameters θ, ρ, and σ, is independent of A and B. Further, the values of A and B may be recovered from the mean μ and the magnitude ν.

The results of applying the Karhunen-Loéve expansion are displayed in Figures 1(c) and 1(d). In Figure 1(c) we display the 8 most important eigenvectors, ordered by their eigenvalues. The similarity between the first 4 eigenvectors and the ones derived in [Hummel 79] is immediate. On closer inspection, however, we notice that while Hummel's eigenvectors are radially symmetric, the ones we computed are not. This is to be expected since the introduction of the parameters ρ and σ serves to break the radial symmetry that Hummel's edge model assumes. While Hummel's eigenvectors are optimal for his edge model, our numerically obtained results imply that they are not optimal for our, more complex, edge model.

The window[3] chosen for our edge model includes 49 pixels. To avoid unnecessary non-linearities induced by a square window, we have used a disc shaped one. In Figure 1(d), the decay of the Karhunen-Loéve residue is plotted as a function of the number of eigenvectors. To reduce the residue to 10% we need to use 3 eigenvectors. To reduce it further to 2% we need 8 eigenvectors. These results illustrate a significant data compression factor of 5-15 times. As a result, feature detection and parameter estimation prove to be efficient. Hummel predicts that for the continuous step edge, the eigenvalues should decay like $1/n^2$. Our results are consistent with this. We found that for our edge model the eigenvalues initially decay like $1/n^2$, but the rate of decay increases with n due to the discretization introduced by realistic modeling of the sensing element.

[3] In our modular implementation, the window may be changed independently of the feature. During our comparison with the Nalwa-Binford and Canny edge detectors we resort to a 5×5 square window for fairness.



(a) Step edge model

(b) Plan view

(c) First 8 eigenvectors

(d) Decay of the K-L residue

(e) Step edge parametric manifold

Figure 1: The step edge model includes two constant intensity regions of brightness A and $A+B$. Its orientation and intrapixel displacement from the origin are given by the parameters θ and ρ, respectively. The fifth parameter (not shown) is the blurring factor σ. The K-L residue plot shows that that 90% of the edge image content is preserved by the first 3 eigenvectors and 98% by the first 8 eigenvectors. The step edge manifold is parameterized by orientation and intrapixel localization for a fixed blurring value and is displayed in a 3-D subspace constructed using the first three eigenvectors.

The step edge manifold is displayed in Figure 1(e). Naturally, we are only able to display a 3-D projection of it into a subspace. This subspace is spanned by the 3 most important eigenvectors. Also, for clarity, we only display a 2 parameter "slice" through the manifold, by keeping σ constant and varying θ and ρ. As mentioned earlier, the first 3 eigenvectors capture more than 90% of the information. This is reflected by Figure 1(e), where most points on the manifold are seen to lie at a unit distance from the origin. Note that the four apparent "singularities" of the manifold are simply artifacts of the projection of the manifold to the 3-D subspace. If we were able to visualize a higher dimensional projection, these would disappear.

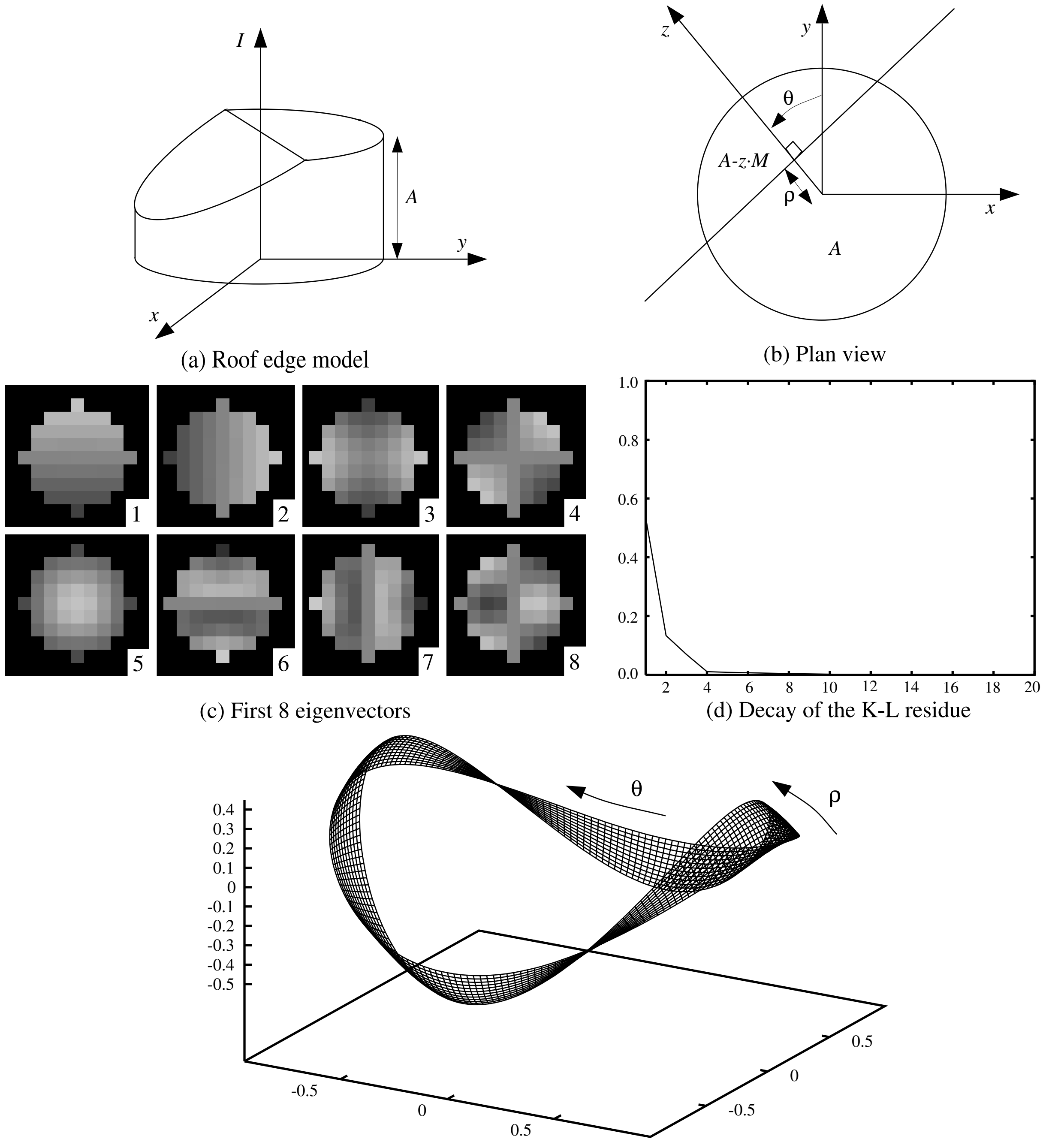
3.2 Roof Edge

The roof edge is in some respects similar to the step edge. However, unlike the step edge, it has not been explored much in the past despite having been acknowledged as a pertinent feature [Nalwa 93]. The main difference between the two edge models is that the step discontinuity is removed from the step edge and the lower intensity step is replaced with a slanting "roof", as shown in Figure 2(a). A formal definition can be obtained by replacing $A + B \cdot u(z)$ with $A - M \cdot z \cdot u(z)$, where A is the upper intensity level of the roof, and M is the gradient, or slope, of the roof. The result is a 5 parameter model written as:

$$F_{RE}(x, y; A, M, \theta, \rho, \sigma) = \{\, (A - M \cdot z \cdot u(z)) * g(x, y; \sigma) * \; a(x, y) \,\} \, . \, s(x, y) \qquad (11)$$

where $u(z)$ and z are as defined for the step edge. The parameter ranges we used are: $\theta \in [0^o, 360^o]$, $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, and $\sigma \in [0.4, 1.0]$. The parameters A and M are free as before. As in the case of the step edge, the structure of the normalized roof edge is independent of A and M, and their values are easily recovered from the normalization coefficients μ and ν.

The results of applying the Karhunen-Loéve expansion, shown in Figures 2(c) and 2(d), are similar to those for the step edge. The K-L residue decays slightly faster in this case. This might be expected since the roof edge more closely resembles a constant intensity region. The residue of a constant intensity region would be exactly zero. Also note that the first two eigenvectors are approximately the same as those for the step edge (at least, up to a rotation of 180^o). 3 eigenvectors are needed to capture 90% of the edge content, and 5 eigenvectors for 98%. The parametric manifold for the roof edge is displayed in Figure 2(e). The radical difference in appearance from the one for the step edge is entirely due to the difference between the third eigenvectors of the two features. The projection onto the first two eigenvectors is similar.



(a) Roof edge model

(b) Plan view

(c) First 8 eigenvectors

(d) Decay of the K-L residue

(e) Roof edge parametric manifold

Figure 2: The roof edge model has on one side a region of constant brightness A and on the other a brightness gradient M. Both these parameters are removed by brightness normalization. The orientation parameter θ, the localization parameter ρ, and the blur parameter σ are similar to those used for the step edge. The first few eigenvectors of roof edge are similar to those of the step edge, but the K-L residue decays marginally faster. The displayed roof edge manifold is parameterized by orientation and intrapixel displacement for a fixed blurring value.

3.3 Line

The line consists of a pair of parallel step edges separated by a short distance, namely, the width w of the line [Hueckel 73] [Lenz 87]. Our line model is illustrated in Figure 3(a). In our definition, we assume that the intensity steps are both of the same magnitude. It is possible, to generalize this model to lines with arbitrary brightness on either side with the addition of one extra parameter [Hueckel 73]. This line model has 6 parameters and is formally expressed as:

$$F_L(x, y; A, B, \theta, \rho, w, \sigma) = \{\, (A+B\cdot u(z+w/2)-B\cdot u(z-w/2)) * g(x, y; \sigma) * a(x, y) \,\} \,.\, s(x, y) \tag{12}$$

The ranges of the parameters ρ and σ are exactly as for the roof edge; $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, and $\sigma \in [0.4, 1.0]$. Given the brightness symmetry in our line model, the orientation range can be halved to $\theta \in [0^o, 180^o]$. We restrict the line width to $w \in [1.0, 3.5]$. In the line model too, the brightness parameters A and B are free and can be removed through the normalization procedures presented in section 2.4. Again, during detection, A and B can be recovered from the normalization coefficients μ and ν.

The result of applying the Karhunen-Loéve expansion is somewhat different from those for the previous features. Most significant is the lower rate of decay in the residue as seen from Figure 3(d). To reduce the residue to 10% we require 8 eigenvectors, and to reduce it to 2% we need 22. By this measure the line is a considerably more complex feature than an edge. However, the data compression factor is still large, and in the range of 3-5. It is interesting to note that the line manifold in Figure 3(e) has the structure of a Möbius band. This results from the following symmetry in the line model:

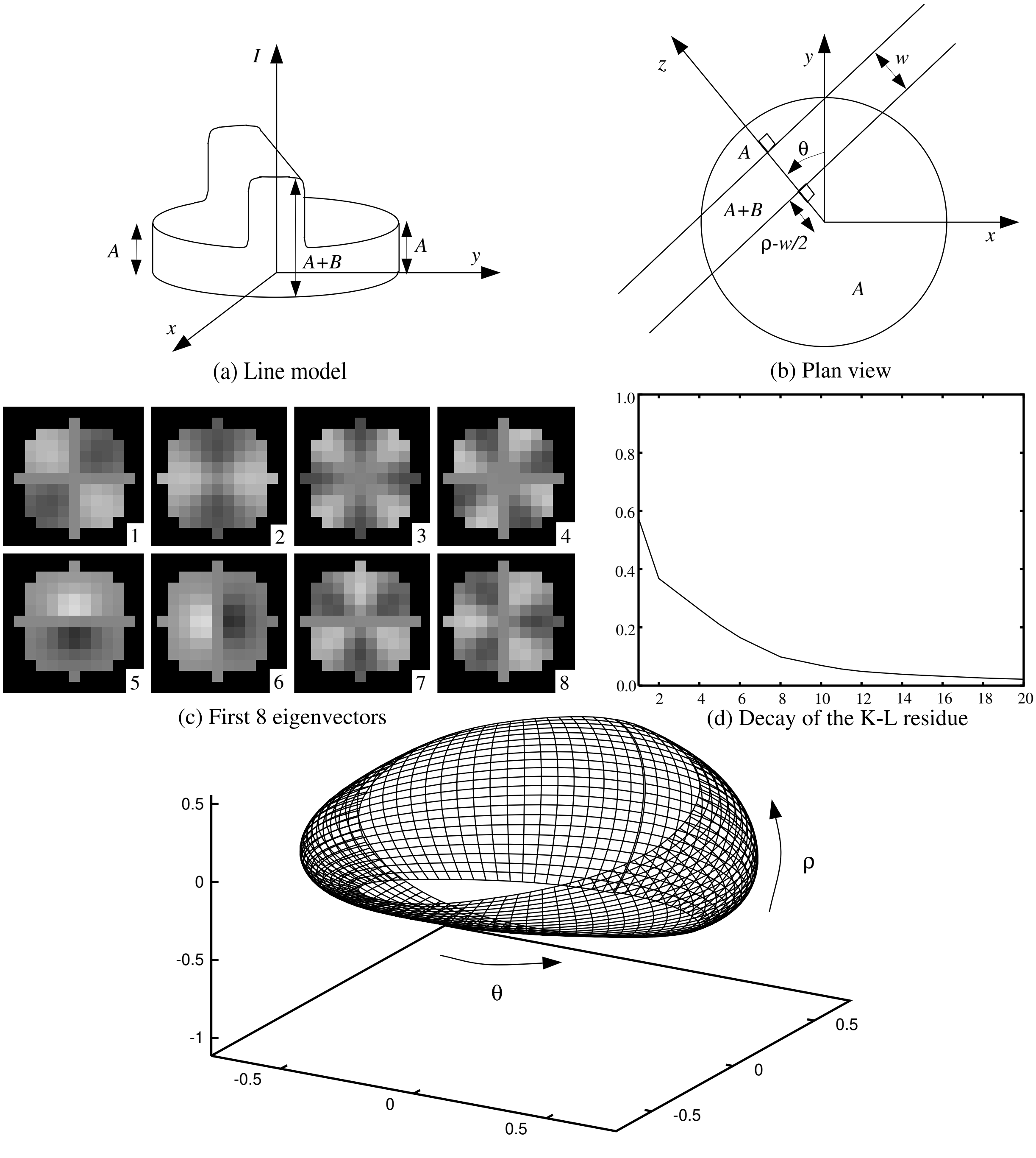
$$F_L(x, y; A, B, \theta + 180^o, \rho, w, \sigma) = F_L(x, y; A, B, \theta, -\rho, w, \sigma) \tag{13}$$

3.4 Corner

The corner is a common and hence important image feature [Nobel 88]. In our corner model, shown in Figure 4(a), θ_1 is the angle of one of the edges of the corner, and θ_2 the angle subtended by the corner itself. The corner lies at the intersection of its edges at angles θ_1 and $180^o + \theta_1 + \theta_2$, as illustrated in Figure 4(b). Mathematically, this intersection can be expressed as the product of two unit step functions. The complete corner model has 5 parameters and is written as:

$$F_C(x, y; A, B, \theta_1, \theta_2, \sigma) = \{\, (A+B\cdot u(z(\theta_1))\cdot u(z(180^o+\theta_1+\theta_2))) * g(x, y; \sigma) * a(x, y) \,\} \,.\, s(x, y) \tag{14}$$

where, $z(\theta) = y \cdot \cos\theta - x \cdot \sin\theta$. The parameter ranges used are: $\theta_1 \in [0^o, 360^o]$, $\theta_2 \in [30^o, 120^o]$, and $\sigma \in [0.4, 1.0]$. Again, brightness normalization eliminates the parameters



(a) Line model

(b) Plan view

(c) First 8 eigenvectors

(d) Decay of the K-L residue

(e) Line parametric manifold

Figure 3: The line is of width w and brightness $A + B$, and has regions of brightness A on both sides. In addition, it has the orientation parameter θ, the localization parameter ρ, and the blur parameter σ. 8 eigenvectors are need to capture 90% of the feature content and 22 eigenvectors for 98%. By this measure the line is a considerably more complex feature than an edge. The line manifold has the structure of a Möbius band.

A and B. The decay of the K-L residue, as in Figure 4(d), is similar to that of the line. In this case, 7 eigenvectors reduce the residue to 10%, and 15 eigenvectors are needed to reduce it to 2%. The corner manifold has a rather interesting shape, as in Figure 4(e).

3.5 Circular Disc

Our final example is the circular disc, illustrated in Figure 5(a) and Figure 5(b). The parameters of the circular disc are its radius r, the direction θ of the center P of the disc from the origin, its localization ρ, and the blurring σ. The brightness values inside and outside the disc are $A+B$ and A, respectively. Formally, the corner model can be expressed as:

$$F_{CD}(x,y;A,B,\theta,\rho,r,\sigma) = (A + B \cdot u(r - \mathrm{d}(x,y))) * g(x,y;\sigma) \tag{15}$$

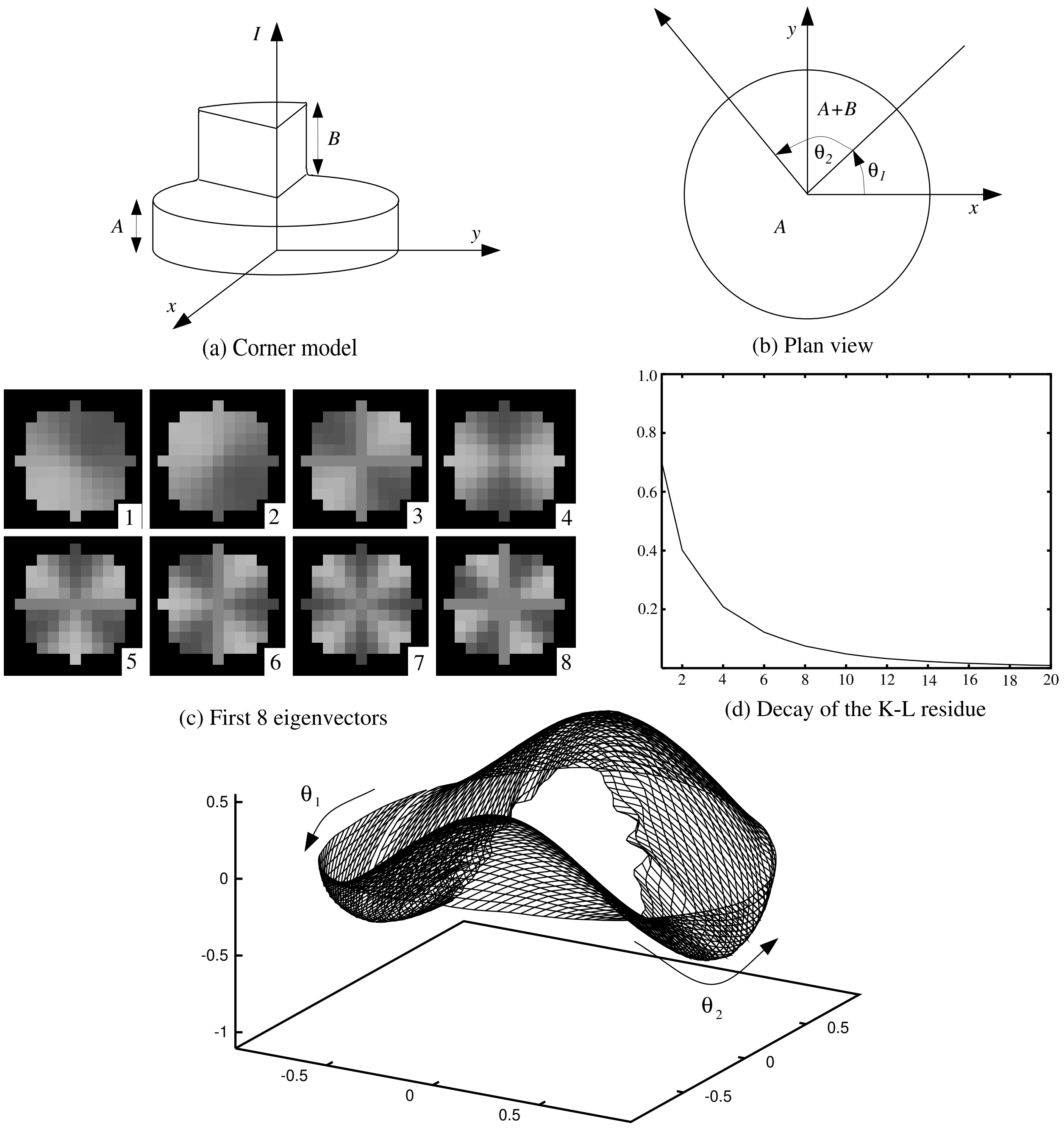
where, $\mathrm{d}(x,y) = \sqrt{((x + (r-\rho)\sin\theta)^2 + (y - (r-\rho)\cos\theta)^2)}$, is the distance of the point, P, from (x,y). The parameter ranges are: $\theta \in [0^o, 360^o]$, $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, $r \in [3.0, 12.0]$, and $\sigma \in [0.4, 1.0]$. Again, brightness normalization removes the effects of A and B. The decay of the K-L residue, as seen from Figure 5(d), is slightly slower than for the step edge. In this case, we need 4 eigenvectors to reduce the residue to 10% of the information, and 11 eigenvectors to reduce it to below 2%. The first 8 eigenvectors are shown in Figure 5(c), and the manifold in Figure 5(e).

4 Feature Detection and Parameter Estimation

We now describe the details of feature detection and parameter estimation. Given a point corresponding to the pixel intensity values in a novel feature window, feature detection requires finding the closest point on the parametric manifold. If the distance between the novel point and the closest manifold point is sufficiently small, we declare the presence of the feature. The parameters of the closest manifold point are then used as estimates of the scene feature's parameters. Alternatively, if the distance between the novel point and the manifold is too large, we assert the absence of the feature. We approximate the closest manifold point by first sampling the manifold, and then performing a search for the closest sample point. So long as we sample densely enough, this yields a sufficiently good estimate of the closest manifold point. In this section, we first discuss how to sample the manifold, and then present our search algorithm. We conclude this section by discussing efficiency.

4.1 Sampling the Parametric Manifold

As we saw in the previous section, after parameter reduction by brightness normalization, we are typically left with a manifold of dimension in the range 2–4. We sample each

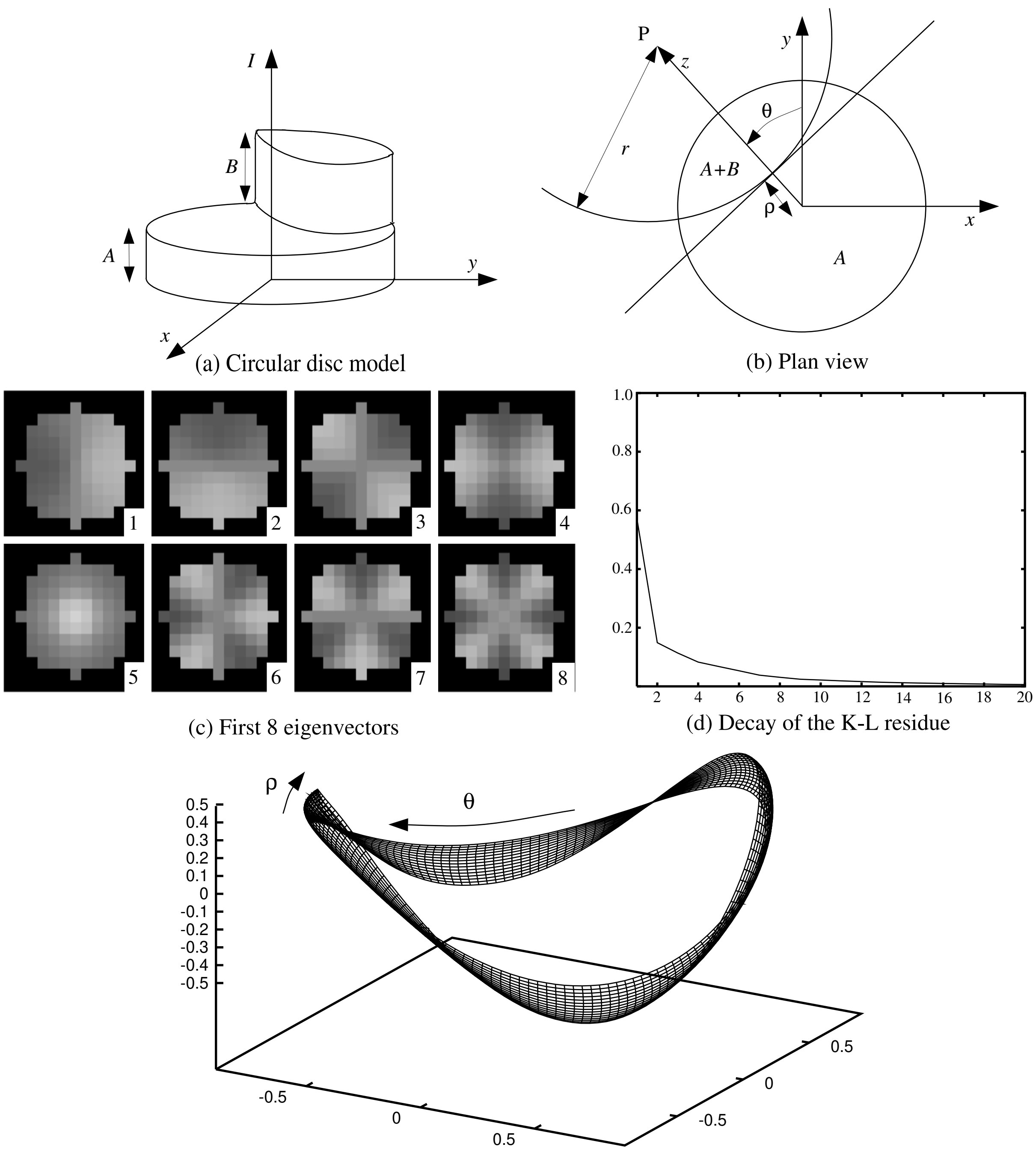


(a) Corner model (b) Plan view

(c) First 8 eigenvectors (d) Decay of the K-L residue

(e) Corner parametric manifold

Figure 4: The corner is described by the brightness values (A and $A + B$) inside and outside the corner, the angles θ_1 and θ_2 made by its edges, and the blur parameter σ. 7 eigenvectors are needed to preserve 90% of the information and 15 eigenvectors for 98%. The corner manifold is shown for a particular value of σ.



(a) Circular disc model

(b) Plan view

(c) First 8 eigenvectors

(d) Decay of the K-L residue

(e) Circular disc parametric manifold

Figure 5: The circular disc is described by the brightness parameters A and B, the radius r of the disc, the angle θ subtended by the center of disc, the localization ρ, and the blur parameter σ. 4 eigenvectors are needed to preserve 90% of the information, and 11 eigenvectors for 98%. The displayed disc manifold is for fixed values of σ and r.

Feature	Step Edge		Roof Edge		Line		Corner		Circular Arc	
No. Points	46368		52250		55440		50196		55440	
	$\Delta\theta$	1.613^{o}	$\Delta\theta$	0.861^{o}	$\Delta\theta$	2.020^{o}	$\Delta\theta_1$	1.351^{o}	$\Delta\theta$	2.337^{o}
	$\Delta\rho$	0.088	$\Delta\rho$	0.081	$\Delta\rho$	0.093	$\Delta\theta_2$	1.819^{o}	$\Delta\rho$	0.139
	$\Delta\sigma$	0.135	$\Delta\sigma$	0.142	$\Delta\sigma$	0.390	$\Delta\sigma$	0.500	$\Delta\sigma$	0.189
					Δw	0.179			Δr	2.144

Figure 6: Automatically generated sampling intervals for the 5 features. The intervals are generated by attempting to ensure that the appearance change between each pair of neighboring sample points is the same, while at the same time trying to limit the total to approximately 50,000 sample points. These figures may be used to assess the importance of each parameter to the model.

parameter independently and uniformly, that is, at equally spaced intervals across its range. Then, the Cartesian product of the parameter sample points is taken and used to sample the manifold. If the dimension of the manifold is k, the result is sampling at a k-dimensional mesh of parameter values. For example, we might sample the step edge's angle θ every 2^o, the localization ρ every 0.1 pixels, and the blurring parameter σ every 0.2 pixels. This leaves one question unanswered: How densely should we sample each parameter?

The answer depends on the effect that varying each parameter has on the appearance of the feature. If a particular parameter causes the appearance (distance traveled on the manifold) to vary rapidly, we should sample it densely in order to capture the full variation in appearance. If varying a parameter results in only a small change in appearance, then there is little use in sampling it densely, since noise in the image will fundamentally limit the accuracy with which we can estimate that parameter anyway. As a rough guideline, we should aim for the change in appearance between two neighboring sample points to be approximately the same as the change in appearance we can expect due to noise.

In practice, estimating the effect of noise may be difficult. Instead we would rather specify the number of sample points that we can afford to use, either for time or space complexity reasons. We then desire the most uniform and dense sampling of the manifold that uses approximately the correct number of samples. In Figure 6, we present the output of a program that automatically estimates the rate of change of appearance with respect to each parameter, and then uses these estimates to derive sampling intervals for each parameter. The input to the program was the request to generate manifold samplings, each containing approximately 50,000 sample points. The output is displayed in a separate column for each feature and consists of the sampling interval determined for each parameter.

Figure 6 may be used to calculate the relative importance of the parameters to the models. We illustrate this with an example, although similar calculations may be done for other pairs of parameters. From the figure, we can see that for the step edge, varying ρ by

0.088 pixels has the same effect as varying θ by 1.613^o. So, assuming that we may scale this linearly, which is a reasonable approximation, varying ρ by 0.71 has a similar effect to varying θ by $1.613 \times \frac{0.71}{0.088} \approx 13^o$. Since, 0.71 is half of the interval through which ρ varies, the importance of ρ to the parametric representation of the step edge model can be seen to be approximately equivalent to 13^o of θ. A similar calculation for σ shows it to be equivalent to approximately 7^o of θ. A quick glance through Figure 6 shows all the parameters to be of high importance to the parametric representations of their respective features. Possible exceptions are the burring parameters of the line and the corner.

4.2 Search for the Closest Manifold Point

The general problem of finding the nearest neighbor to a given novel point in a high dimensional space is well studied in computational geometry. The paper by Yinailos [Yianilos 93] contains a comprehensive bibliography of work on nearest neighbor search. Rather than using any of the general purpose techniques mentioned in [Yianilos 93], we attempt to take advantage of the properties of the feature manifolds and develop a less general but faster search technique. We used a heuristic coarse-to-fine search, which relies upon, and takes advantage of, the locally smooth behavior of the feature manifolds. The search does not guarantee finding the closest point for pathological manifolds, but we found empirically that is performs very well for the 5 example features. In particular, for the manifolds sampled at 50,000 points, the heuristic search results in a speed-up by a factor of 50-100 times, over linear search through the 50,000 points.

The coarse-to-fine search is conceptually very simple. We sample the manifold several times, giving a sequence of meshes, from a very coarse one up to the finest one. The finest mesh is the one we really wish to search. We begin by finding the closest point on the coarsest mesh by using a brute force linear search. This does not cost much in terms of time since the coarsest mesh does not contain many points. We then move to the next finer mesh. We search this mesh locally in the region of the result of the previous level. This search is also a linear brute force search. It again does not cost much since it is only a local search, and on a relatively coarse mesh. We repeat this for each mesh in turn, reducing the size of the local search at each step, until we reach the finest mesh. The result at the finest search gives us the answer we are looking for.

4.3 Further Efficiency Improvements

On a standard single-processor workstation with no additional hardware, the coarse-to-fine search for a 3-D manifold in a 10-D space that is sampled at 50,000 points, takes approximately 1ms. So, applying to every pixel in a 512×480 image takes around 4mins. Even this figure is not totally unreasonable for some applications, however it is by no means the best we can do in terms of efficiency:

Rejection We do not need to apply the coarse-to-fine search at every pixel in the image. The idea of doing this is as old as edge detection itself, and is explicitly mentioned in [Hueckel 71]. Combining a variety of techniques, we have already reduced the time to process a 512×480 image to less than a minute. In particular, we currently threshold on the magnitude, ν, obtained during normalization. This approach is similar to Moravec's interest operator [Moravec 77], used to predict the usefulness of stereo correspondence matches. We also threshold on the distance from the K-L subspace. Since, this is (approximately) a lower bound on the distance from the manifold, if the distance from the K-L subspace is too large, we can immediately decide that the pixel does not contain the feature, and so avoid the search. Using the techniques in [Baker and Nayar 95], we can even avoid most of the cost of computing the distance from the K-L subspace. The distance from the K-L subspace has been used in the past for various classification purposes, for example in the field of face recognition [Pentland et al. 94], where it is termed the *distance from face/feature space.*

Parallel Implementation Feature detection is inherently a parallelizable task. As high performance multi-processor workstations become commonplace, the times mentioned above may easily be cut by factors on the order of 10 or more. Also, it is reasonable to expect performance increases for the individual processors, further increasing the overall performance. Real-time performance, already achievable on todays fastest machines, will be possible on desktop machines within a few years.

5 Experimental Results

Assessing the performance of a feature detector is rarely addressed in a satisfactory manner. Of the papers that do comprehensively investigate detector performance, we strongly recommend both [Nalwa and Binford 86] and [Abdou and Pratt 79]. The monograph by Pratt [Pratt 90] also contains a thorough discussion of proposed techniques. We highlighted 4 ways to evaluate the performance of feature detectors:

1. Statistically compare the rates of occurrence of false positives and false negatives.
2. Compare the accuracy of parameter estimation also using statistical tests.
3. Subjectively compare detector outputs applied to real and/or synthetic images.
4. Compare measures that combine feature detection rates with parameter estimation accuracy. One example is Pratt's Figure of Merit [Pratt 90].

In the next three subsections, we explore the first three methods in turn. We chose not to investigate any composite measures, even though we favor the use of standard performance tests. We wish to emphasize the distinction between detection and parameter estimation, and so treat each task independently.

5.1 Feature Detection Rates

We begin by statistically comparing our step edge detector with both the Canny [Canny 86] and Nalwa-Binford [Nalwa and Binford 86] detectors. A totally fair comparison constitutes a very difficult undertaking and warrants a detailed study in itself. For reasons of consistency with previous work, we follow the approach in [Nalwa and Binford 86].

The first difficulty is that, each detector is based on its own model of an edge. Our model and the Nalwa-Binford model are closely related, but comparison with the differential invariant based Canny operator is problematic. Since we took great care modeling both the features and the imaging system, we used our step edge model in the comparison. For fairness however, we changed the details slightly. Both the Canny and Nalwa-Binford detectors assume a constant blur/scale, so we fixed the value of σ in the step edge model to be 0.6 pixels. Secondly, the Nalwa-Binford detector is based on a square 5×5 window, as is Canny for the implementation that we used. Hence, we changed the window of our detector to be a square window containing $N = 25$ pixels, rather than the 49 pixel disc window used earlier.

Another issue is the lack of a model for a characteristic "not edge" [Nalwa 93]. Whereas it is simple to generate ideal edges, add noise, and then apply the detector to estimate false positives, generating not edges that we can use to estimate false positives requires some model of a not edge. We resolve this, as in [Nalwa and Binford 86], by taking a constant intensity window as our not edge, and then adding white zero-mean Guassian noise. After the normalizations in subsection 2.4, this is (almost) equivalent to picking a "not edge" uniformly at random from the surface of the unit sphere in the N-dimensional Hilbert Space, upon which the edge manifold lies [Knuth 81].

In Figure 7 we compare the detection performance of the three edge detectors. For each pair of S.N.R. and detector, we plot a curve of false positives against false negatives obtained by varying the threshold inherent in each detection algorithm. The Canny operator thresholds on the gradient magnitude, the Nalwa-Binford detector thresholds on the estimated step size, and our approach thresholds on the distance from the parametric manifold. As described before, the rate of false positives is obtained by applying the detector to a constant intensity window with noise added. The rate of false negatives is obtained by applying the detectors to noisy step edges.

The closer a curve lies to the origin in Figure 7, the better the performance. Hence, we can see that both the Canny detector and our detector do increasingly well as the S.N.R. increases. In fact, for low levels of noise the Canny detector does marginally better than our technique. There are a number of potential explanations for this. One possibility is that when we generate constant intensity windows with a low level of noise, there is a non-zero

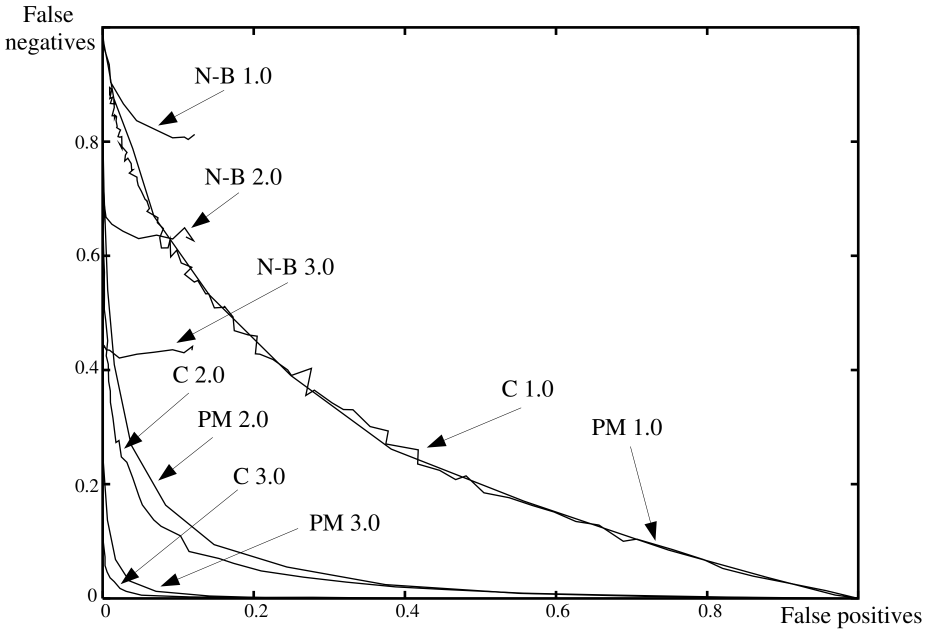


Figure 7: A comparison of edge detection rates. The Canny (C), Nalwa-Binford (N-B), and parametric manifold (PM) detectors are compared for S.N.R. = 1.0, 2.0, and 3.0. We plot false positives against false negatives. For each detector and S.N.R., the result is a curve parameterized by the threshold inherent in that detector. The closer a curve lies to the origin, the better the performance. We see that the Canny detector and the parametric manifold technique perform similarly, with the Canny detector doing marginally better for low levels of noise. The results for the Nalwa-Binford detector (which are consistent with the results presented in [Nalwa and Binford 86]) are completely different.

probability that the result is approximately a step edge, but with a small value for the step parameter B. Due to our normalization procedure, such an input will be detected as an edge by our scheme, but not by Canny since the step size is too small. Strictly speaking, in terms of our model this is not a false positive, but is registered as such in our experimental results. If our reasoning is correct, we would expect to see Canny doing increasingly better than our technique as the noise level is reduced. This is consistent with Figure 7.

Our results for the Nalwa-Binford detector are consistent[4] with those described in [Nalwa and Binford 86]. Applied to real images, the Nalwa-Binford detector does not perform as poorly as Figure 7 might indicate. This highlights the difficulty in statistically comparing edge detectors. The poor Nalwa-Binford results are probably due to thresholding on the step-size. They may well be completely different if we fix the step-size threshold,

[4] We did use step 2)′ of the Nalwa-Binford algorithm. The inclusion of this step does not radically alter the performance. See [Nalwa and Binford 86] for a discussion of the exact details of how step 2)′ effects the results.

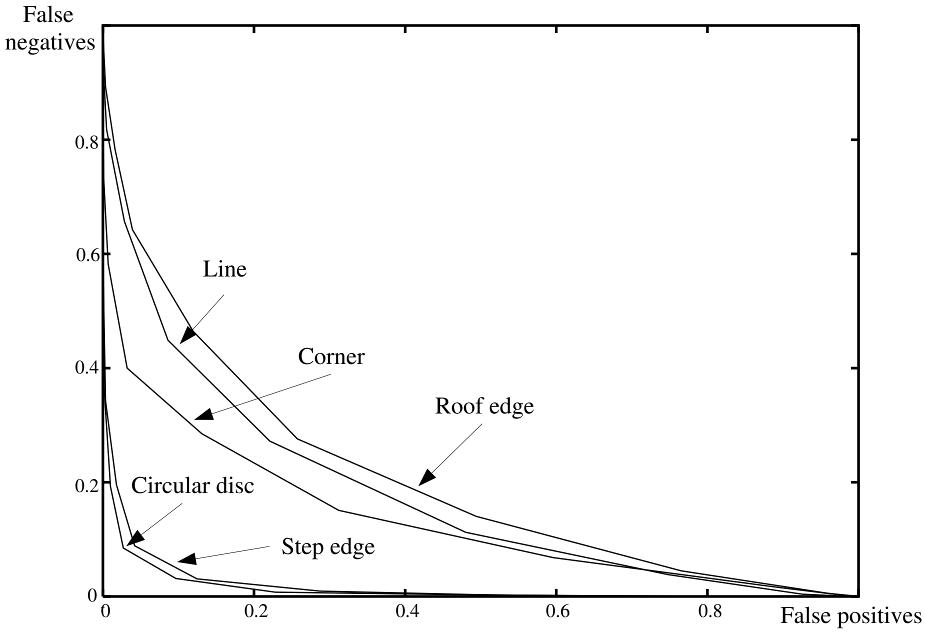


Figure 8: A comparison of feature detection rates for our 5 example features. All results are for S.N.R. = 1.5 and for a disc shaped window containing 49 pixels. We see that the step edge and circular disc are less noise sensitive than the other features. Note however that the noise sensitivity of all features may be reduced by increasing the size of the window. Further, for S.N.R. around 3.0 and above, all the feature detectors perform with very little error. Hence, we claim success in our aim of finding a general purpose approach to parametric feature detection.

and vary a tanh-fit threshold. Further investigation is outside the scope of this paper.

Next, in Figure 8 we compare the detection rates of our 5 example features. The rates of occurrence of false positives and false negatives are estimated in exactly the same method as above. In the figure, the plots are all for a S.N.R. of 1.5, and for a disc shaped window comprising 49 pixels. We see that the performances of the step edge and the circular disc are superior to that of the other 3 features. Since the definition of the S.N.R. is radically different for the roof edge, little should be read into its relatively poor performance. We do conclude, however, that the corner and line are more sensitive to noise than the other features. One method of reducing the noise sensitivity is to use a slightly larger window. If we increase the window size to a disc containing 81 pixels, the performance is greatly enhanced. We also found that the performance of each of the 5 feature detectors improves with the S.N.R. exactly as it did for the step edge in Figure 7. For S.N.R. ≈ 3.0 and above, all the detectors perform almost without error. From these (unshown) results for medium to low levels of noise, we conclude success in our goal of developing a feature detection technique applicable to arbitrary parameterized features.

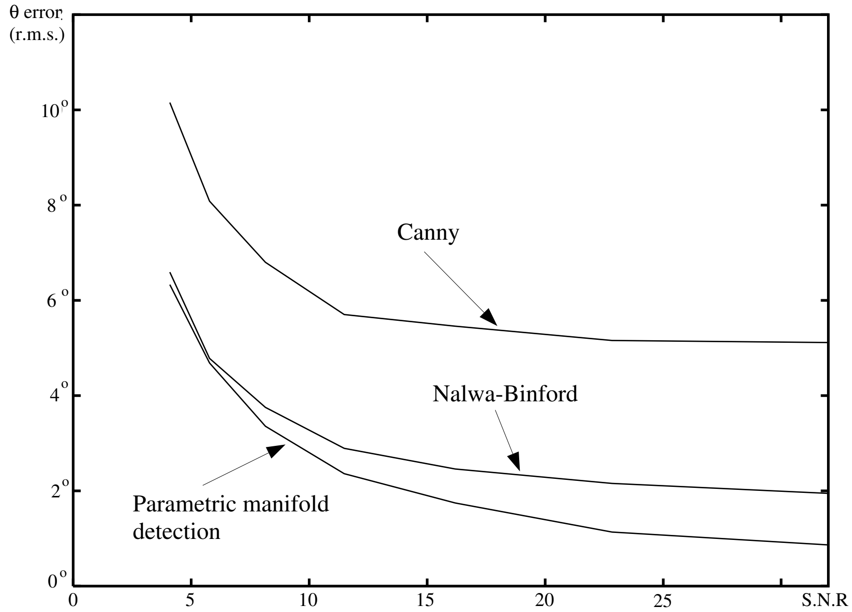


Figure 9: A comparison of edge detector orientation estimation accuracy. We took a synthesized step edge, added noise to it, and then applied the edge detectors. We plot the r.m.s. error of the orientation estimate against the S.N.R. We can see that for high levels of noise (low S.N.R.) the accuracy is limited by the noise. As the noise level decreases, the parametric manifold approach outperforms both the Nalwa-Binford and Canny detectors.

5.2 Parameter Estimation Accuracy

Compared to feature detection robustness, assessing the performance of parameter estimation is relatively straightforward. Generalizing the procedure in [Nalwa and Binford 86], we randomly generate a set of feature parameters, synthesize a feature with these parameters, add a certain amount of noise, apply the detector, and then measure the accuracy of the estimated parameters. If we repeat this procedure a statistically meaningful number of times, the results should give a very good indication of parameter estimation performance when applied to a real image. The issue of which model we should use to generate the features is still somewhat problematic. However, for the same reason as before, we will use our feature models to generate the synthetic features.

In Figure 9 we compare the performance of our step edge detector with that of the Canny detector [Canny 86] and the Nalwa-Binford [Nalwa and Binford 86] detector. For fairness, as above we used the parametric step edge detector computed for a 5×5 square window, and with the blurring parameter fixed at 0.6 pixels. In the figure, we plot the r.m.s. error in the estimate of the orientation θ against the S.N.R. The plot is consistent with the performance figures for the Nalwa-Binford detector presented in [Nalwa and Binford 86]. We see that for low S.N.R. the performance of all detectors is limited by the noise. For

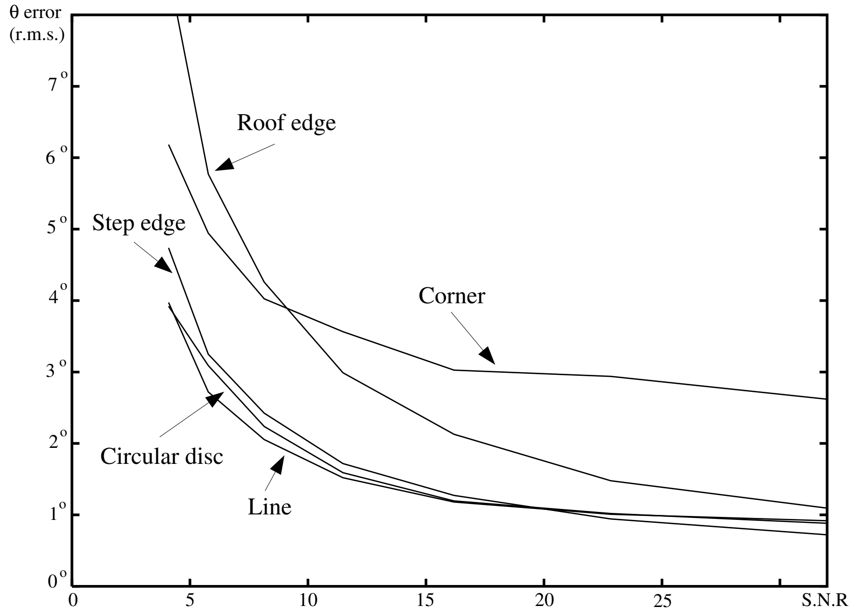
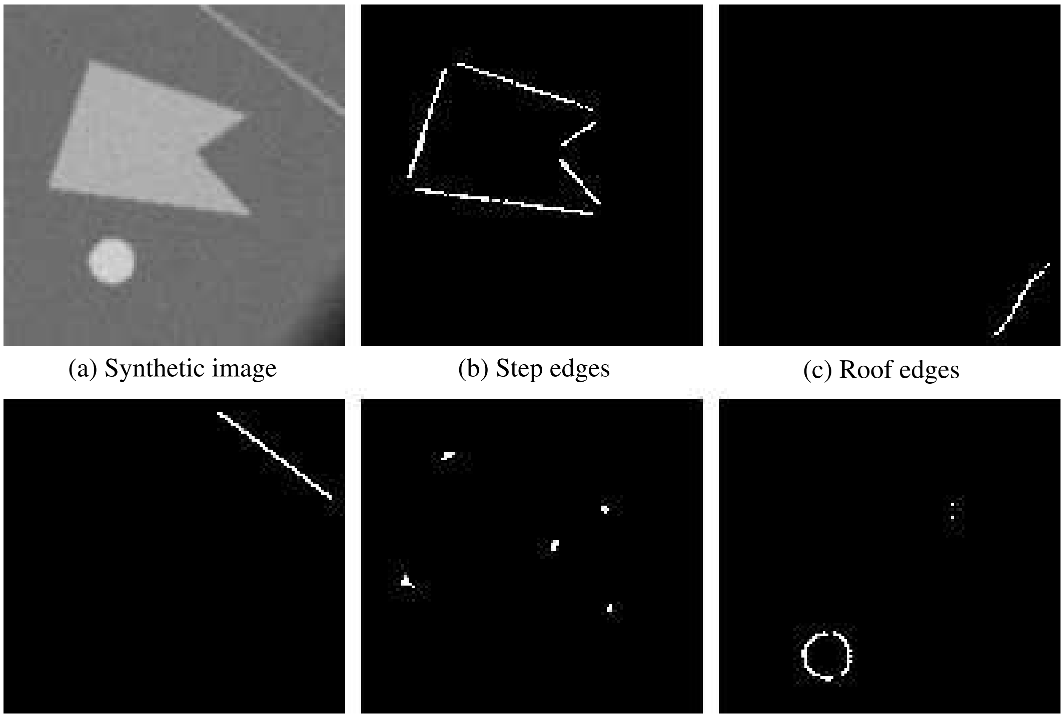


Figure 10: A comparison of the examples features. Since all 5 features have an orientation parameter, we use it to compare the performance. We plot the (r.m.s.) orientation estimation error against the S.N.R. The graph shows that the performance for all 5 features is approximately the same. From this we conclude success in our goal of developing a parameter estimation technique applicable to general parametric features.

low noise levels, our detector is only restricted by how densely the manifold is sampled. Hence, it out performs both of the other detectors. If we plot a similar graph for the parameter ρ, we find the behavior to be similar. In this case however, the performance of the Nalwa-Binford detector and the parametric manifold technique are very similar, but with the Nalwa-Binford detector doing marginally better.

Next we compare the performance of our 5 features. Since all the feature models have an orientation parameter, in Figure 10 we plot the r.m.s. error in orientation error against the S.N.R. Note that, as before direct comparison of the roof edge results with those for the other features is somewhat difficult due the different definition of the S.N.R. From Figure 10, we see the behavior to be qualitatively the same for all the features. Further, the graphs for the line, step edge, and circular disc are almost identical. Only for the corner and roof edge is the decay in parameter estimation accuracy slightly slower. Although we do not have space to include them, the plots of estimation accuracy for the other parameters (including those involved in normalization) are all very similar. From this, we conclude success. The parametric manifold technique yields parameter estimation accuracy, comparable with the best available edge detection techniques, but for arbitrarily defined parametric scene features.



(a) Synthetic image (b) Step edges (c) Roof edges

(d) Lines (e) Corners (f) Circular discs

Figure 11: The application of our 5 feature detectors to a synthetic image. For the first time, we are able to completely detect and discriminate all 5 example features in the same image using the same technique.

5.3 Application to images

In Figures 11(b)–(f) we display the results of applying the 5 example detectors to the synthetic image in Figure 11(a). The synthetic image is of size 128×128 pixels and contains a pentagonal region (intensity 175), a circular disc (radius 8.5 pixels, intensity 206), a line (width 2.3 pixels, intensity 153), and a roof edge (slope 4 intensity levels per pixel). The background intensity is 110. The image was first blurred with Guassian smoothing ($\sigma = 0.6$ pixels) and then we added white zero-mean Gaussian ($\sigma = 4.0$ pixels) noise. At pixels where two feature detectors both register the presence of a feature, we choose the one with the closer manifold. As can be seen from Figure 11, all of our detectors perform very well. For the first time, as far as we are aware, we are able to detect in a single image 5 different features. Further, the technique may easily be generalized to other user-defined parametric features.

We further demonstrate the versatility of our parametric manifold technique by applying the corner detector to a real image. In Figure 12, we have superimposed the output of the corner detector on top of the image to which it was applied. The image is of size 401×267 pixels, which the corner detector took 24.4 seconds to process. As can

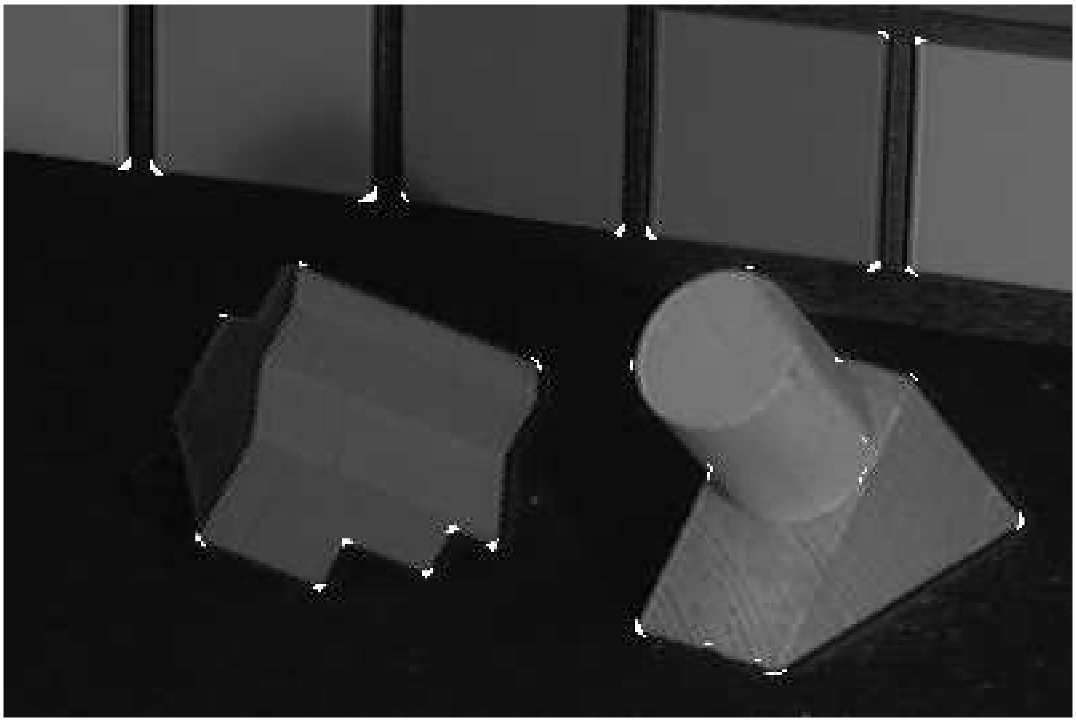


Figure 12: The corner detector applied to a real image. We have reduced the intensity of the original image, and then superimposed the detected corners as high intensity points. The parametric corner detector can be seen to perform very well, detecting all the corners of angle within the correct range. The processing time for the 401×267 image was 24.4 seconds.

be seen in the figure, the results are very good.. Almost all of the corners are detected. The corners missing on the left hand object are not detected only because their angles are greater than 120^o. Remember that in our parametric definition of a corner, the range of the angle subtended by the corner was $[20^o, 120^o]$. There are some false positives on the other object. A couple lie on the circular disc. Discriminating circular discs and corners is a difficult task due to the inherent similarity in their appearance for larger radius discs and higher angle corners. There are a couple of other false positives lying in the body of the right hand object, caused by the texture of the wood that the object is made from. Two other corners are detected on the "shoulder" of the right object, which have angles slightly greater the 120^o.

6 Discussion

What we have put forth is a general mechanism for generating detectors of parameterized features. The techniques we have developed are modular in nature, allowing a user to design a detector for a new feature by simply defining its model. This is in strong contrast to previous approaches to feature detection where each feature was treated as a separate visual entity that deserves its own detailed analysis. Our detailed modeling of features and the artifacts induced by the imaging system have enabled accurate estimation of feature parameters. Though the five features we experimented with are well-known ones, in principle, any parametrized feature lies within the scope our framework. Given the generality and performance of our detectors, we have initiated the development of a comprehensive software package that would allow detector design and application with minimal user interaction.

When multiple detectors are applied to an image, the result is a rich description of the scene. However, to extract high-level primitives from this description, such as, continuous lines and curves, the feature parameters need to be further analyzed. This could be achieved using a relaxation scheme [Rosenfeld et al. 76]. While previous relaxation algorithms have assumed a single feature in the image, often the edge, powerful constraints result from the use of multiple feature detectors. For instance, a corner cannot exist in isolation and must have edges in its vicinity. Multi-feature relaxation could turn out to be an interesting problem.

From a computational perspective, all features are dealt with in essentially the same manner in our detection scheme. The normalizations, subspace projections, and search techniques are consistent across features. This allows us to explore a parallel multi-processor hardware architecture where each processor is dedicated to the detection and parameter estimation of a single feature. Then, the addition of a new feature detector would only involve the addition of a single processor to the architecture. A relaxation processor would take the outputs of all the detection processors to generate high-level primitives. We hope to pursue the implementation of such a feature detection machine.

References

[Abdou and Pratt 79] I.E. Abdou and W.K. Pratt, "Quantitative Design and Evaluation of Enhancement/Thresholding Edge Detectors," *Proceedings of the IEEE,* 67:753–763, 1979.

[Baker and Nayar 95] S. Baker and S.K. Nayar, "A Theory of Pattern Rejection," *Columbia University Technical Report,* CUCS-013-95.

[Barbe 80] D.F. Barbe, editor, *Charge-Coupled Devices,* Springer-Verlag, 1980.

[Born and Wolf 65] M. Born and E. Wolf, *Principles of Optics,* Permagon Press, 1965.

[Bracewell 78] R.N. Bracewell, *The Fourier Transform and Its Applications,* Second Edition, McGraw-Hill Book Co., 1978.

[Canny 86] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[Davis 75] L.S. Davis, "A Survey of Edge Detection Techniques," *Computer Graphics and Image Processing,'* 4:349–376, September 1975.

[Fukunaga 90] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* Academic Press, 1990.

[Haralick 84] R.M. Haralick, "Digital Step Edges from Zero Crossing of Second Directional Derivatives," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 6:58–68, 1984.

[Hueckel 71] M.H. Hueckel, "An Operator Which Locates Edges in Digitized Pictures," *Journal of the Association for Computing Machinery,* 18:113–125, 1971.

[Hueckel 73] M.H. Hueckel, "A Local Visual Operator Which Recognizes Edges and Lines," *Journal of the Association for Computing Machinery,* 20:634–647, 1973.

[Hummel 79] R.A. Hummel, "Feature Detection Using Basis Functions," *Computer Graphics and Image Processing,* 9:40–55, 1979.

[Knuth 81] D.E. Knuth, *The Art of Computer Programming, Volume II: Seminumerical Algorithms*, Addison-Wesley, 1981.

[Koenderink 84] J.J. Koenderink, "The Structure of Images," *Biological Cybernetics,'* 50:363–370, 1984.

[Lenz 87] R. Lenz, "Optimal Filters for the Detection of Linear Patterns in 2-D and Higher Dimensional Images," *Pattern Recognition,* 20:163–172, 1987.

[Moravec 77] H.P. Moravec, "Towards Automatic Visual Obstacle Avoidance," In *Proc. of the Fifth Internation Joint Conference on Artificial Intelligence,* pp. 598–600, 1977.

[Murase and Nayar 95] H. Murase and S.K. Nayar, Visual Learning and Recognition of 3-D Objects from Appearance," *International Journal of Computer Vision,* 14:5–24, 1995

[Nalwa 93] V.S. Nalwa, *A Guided Tour of Computer Vision*, Addison-Wesley, 1993.

[Nalwa and Binford 86] V.S. Nalwa and T.O. Binford, "On detecting edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:699–714, 1986.

[Nobel 88] J.A. Nobel, "Finding corners," *Image and Vision Computing,* 6:121–127, 1988.

[Norton 82] H.N. Norton, *Sensor and Analyzer Handbook,* Prentice-Hall, 1982.

[Oja 83] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press, 1983.

[Pentland et al. 94] A.P. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 84–91, 1994.

[Pratt 90] W.K. Pratt, *Digital Image Processing,* John Wiley & Sons, 1990.

[Rosenfeld et al. 76] A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Transactions on Systems, Man, and Cybernetics,* 6:420-433, 1976.

[Yianilos 93] P.N. Yianilos, "Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces," *Proc. of ACM-SIAM Symposium on Discrete Algorithms,* pp. 311–321, 1993.